Deep Learning (CSE641/ECE555)
Mid-Sem (20 Marks) (Duration 60 min)

Name ......
Roll No. ......

Beginner: 10 Marks

1. Can we use transposed convolution with fractional strides. If so, how does using a fractional stride (e.g., 0.5) in a transposed convolution layer with appropriate padding affect the output size compared to using an integer stride? 1 Mark B

   (a) No, Code implimentation not possible (b) Yes, Increases the output size compared to using the same integer stride. (c) No, Does not affect output size compared to using the same integer stride. (d) Yes, Introduces checkerboard artifacts in the output due to the non-integer stride placement.

2. Consider a neural network with an input layer, one hidden layer, and an output layer. The input layer has 3 neurons, the hidden layer has 5 neurons, and the output layer has 1 neuron. If all the activation functions are sigmoid functions, what is the maximum number of linear hyperplanes the network can create? 1 Mark C

   (a) 3 (b) 5 (c) 20 (d) 21

3. The β-VAE is a variational autoencoder that incorporates a hyperparameter β. How does β affect the model's behavior? A

   (a) Higher β encourages latent representation with higher entropy. (b) Lower β leads to poor reconstruction accuracy. (c) β controls convergence of VAE. (d) β controls latent representations dimensions.

4. A VAE with a standard normal prior ($N(0,1)$) for the latent variables is trained on a dataset. The encoder approximates the posterior distribution with a diagonal covariance matrix $\Sigma = diag(\sigma_1^2, \sigma_2^2, ..., \sigma_d^2)$ for the d-dimensional latent space. The diagonal elements σ_i^2 represent the variances on each dimension. If the mean vector of the approximate posterior is $\mu = (0, 0, ..., 0)$, what is the KL divergence between the approximate posterior and the prior for a single data point? A

   (a) KL divergence $= 0.5 * sum(\sigma_i^2)$ (b) KL divergence $= 0.5 * sum(log(\sigma_i^2))$ (c) KL divergence cannot be determined without the data point's reconstruction error. (d) KL divergence depends only on the prior and not the approximate posterior.

5. Gradient penalty ($\lambda * ||\nabla D(x)||_2^2$) is sometimes added to the Wasserstein GAN (WGAN) loss function. What is the purpose of this term in the equation? A

   (a) It enforces a Lipschitz constraint on the discriminator to improve training stability. (b) It encourages the discriminator to learn a specific decision boundary. (c) It directly measures the difference between real and generated data distributions. (d) It enforces a Lipschitz constraint on the generator to improve reconstruction accuracy.

6. What is wrong with this code snippet? 1 Mark

```
x = torch.ones(3, requires_grad=True)
y = x + 2
y.backward()
```

   y is not a scalar, so backward() needs an argument.

7. Gopi wants to use Xavier initialization on a network with n_{in} input and n_{out} output neurons with ReLU activation function. Write the initialization formula for the weights and explain it. 1 Mark

   DL Lecture 7 slide 32

$$\mathcal{U}(-sqrt(6/(n_in + n_out)), sqrt(6/(n_in + n_out)))$$

8. What makes the approximation capabilities of neural networks different than something like, say, Fourier series? 1 Mark

   Non-linear DNN can approximate any function due to nonlinearity, but Fourier can't when certain conditions are not fulfilled, such as discontinuities. DNN has a better approximation rate, given a carefully designed architecture.

9. Why does the activation function for a single hidden layer MLP have to be non-polynomial? 2 Mark

   If the activation function is polynomial, the entire network (input to output mapping) collapses into a polynomial function of the inputs. The Universal Approximation Theorem (Cybenko, 1989; Hornik, 1991) states that a single hidden-layer MLP with a nonlinear activation function can approximate any continuous function on a compact domain arbitrarily well. However, this theorem does not hold if the activation function is a polynomial. The reason is that polynomials are not dense in the space of continuous functions—meaning that some continuous functions cannot be approximated by polynomials. A well-known counterexample is the indicator function of a set, which cannot be expressed using polynomials due to Runge's phenomenon (oscillatory behavior of polynomial approximations).

Intermediate: 6 Marks

1. Grad-CAM++ is an improvement over Grad-CAM visualization technique that provides better localization by considering higher-order gradient terms. The Grad-CAM++ activation map corresponding to the importance weight α_k^c for feature map A_k is computed as:

$$\alpha_k^c = \sum_{i,j} w_k^{ij} \frac{\partial^2 f^c}{\partial (A_k^{ij})^2} \qquad w_k^{ij} = \frac{\frac{\partial f^c}{\partial A_k^{ij}}}{\sum_{p,q} \frac{\partial f^c}{\partial A_k^{pq}}} \qquad L^c_{\text{Grad-CAM++}} = \text{ReLU}\left(\sum_k \alpha_k^c A_k\right)$$

   where w_k^{ij} is the pixel-wise weighting factor.

   Answer the following with explanation:

   (a) Why is ReLU Necessary?

   (b) What happens when the class scores f^c does not change significantly with respect to variations in the feature maps?

   (c) Will GradCAM++ be better than GradCAM if the model's gradients are noisy or unstable?

   (d) Discuss the impact of depth and width on GradCAM++ visualization?

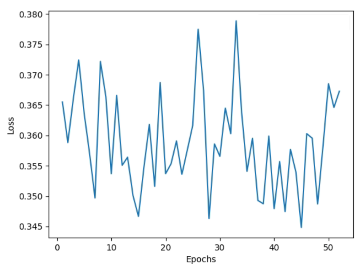
   The negative values in Grad-CAM correspond to regions where decreasing the activation would increase the class score. These regions do not contribute positively to the class and should be ignored in visualization. Applying ReLU ensures that only positive activations are highlighted in the heatmap.
   If the class score does not change significantly with respect to variations in the feature maps, second-order gradients will be close to zero, making Grad-CAM++ ineffective.
   Grad-CAM++ relies on second-order gradients, which are more sensitive to noise. If the model has unstable gradients (e.g., due to poor training or adversarial examples), Grad-CAM++ may highlight irrelevant regions. Example: In adversarially perturbed images, where small pixel changes drastically alter predictions, Grad-CAM++ might amplify the noise rather than provide meaningful localization.
   The second-order gradients become unstable due to vanishing gradient problems in deep layers. Layer aggregation makes it hard for Grad-CAM++ to correctly attribute relevance, leading to diffused or incorrect visualizations. In very wide architectures Grad-CAM++ suffers from gradient saturation across a large number of feature maps, which might cause the attention map to become too spread out, leading to less focused visualizations. Grad-CAM++ needs diverse activation maps to properly distribute importance. In shallow networks, feature maps are too simple, and higher-order gradients become nearly redundant.
   Shallow or extremely deep networks? $\rightarrow$ Use Grad-CAM Moderate-depth models with diverse feature maps? $\rightarrow$ Use Grad-CAM++

2. What is the effect of adding a Laplacian prior to the parameters of the model? Derive the expression of the log-MAP estimate of θ, assuming a Gaussian prior over outputs and Laplacian prior on parameters. 2 Marks Lecture7

Advanced: 4 Marks

1. Parvati is training a ResNet-101 model for image classification, and her training loss graph appears as shown in the figure on the left. Identify the cause of this behavior and suggest remedies in the context of: 2 Marks

   (a) Model Architecture (b) Optimizer



- **Model:** Causes:- Improper Initialization and ineffective Batch Normalization
- **Remedies:**
  - Improper initialization can lead to very high or very low activation values, which in turn can cause the gradients to either explode or vanish. Thus initialization helps maintain a balanced variance throughout the network, promoting stable gradient propagation which reduces oscillations.
  - Batch Normalization is critical for stabilizing the training of networks like ResNet-101. However, if BatchNorm is improperly configured (e.g., due to a very small batch size, running statistics management, learnable (affine transformation)), the computed statistics may be noisy, which can lead to oscillating behavior in the training loss.
- **Optimizer:** Causes:- High Learning rate η
- **Remedy:** If learning rate η in optimizer is set too high the update step becomes excessively large, causing the parameters to overshoot the optimal value which manifests as oscillations in the training loss rather than a smooth descent.

2. In the original GAN formulation, the generator is trained by minimizing $L_G = \mathbb{E}_{z\sim p_z}\left[\log\left(1 - D(G(z))\right)\right]$. Show that when the discriminator D is near optimal, the gradient of the generator's objective can vanish. Then, explain why the alternative objective $L'_G = \mathbb{E}_{z\sim p_z}\left[\log D(G(z))\right]$ is used in practice to mitigate this problem. 2 Marks

   In the original GAN formulation, the generator is trained by minimizing the loss function

$$L_G = \mathbb{E}_{z\sim p_z}\left[\log\left(1 - D(G(z))\right)\right]$$

   where $D(G(z))$ represents the discriminator's output for a generated sample $G(z)$. When the discriminator is near optimal that is, when it can confidently distinguish between real and fake samples the value $D(G(z))$ for generated samples becomes very small (close to 0). As a result, the function $\log\left(1 - D(G(z))\right)$ approaches $\log(1) = 0$, and its derivative with respect to the generator's parameters becomes very small. This phenomenon is known as the vanishing gradient problem. To mitigate this issue, an alternative objective is used for training the generator:

$$L'_G = \mathbb{E}_{z\sim p_z}\left[\log D(G(z))\right]$$

   which is equivalent to minimizing $-\mathbb{E}_{z\sim p_z}\left[\log D(G(z))\right]$. In this alternative formulation, if $D(G(z))$ is very small, then $\log D(G(z))$ becomes a large negative number. Consequently, the gradient with respect to the generator's parameters is amplified, providing a stronger signal that helps the generator improve more effectively during training.